**Purpose:** EPS – Event Processing Subsystem. Provide a base class used by classes like Navigation and Actions. This class manages the task control aspects common to classes that process actions, like queueing events, processing through the current event, etc.

|  |  |
| --- | --- |
| **Priority:** High | **Reason:** Needed as the framework for the classes that will implement it |

**Primary Programmers:**

1. Bryan B
2. Holly L

**Public Constants (public static final):**

* + None

**Constructors (called when an object instance is created):**

* public EPS(0 arguments)
  + Must Initialize: The LLQ (“Low-Level Queue”), which holds all events to be processed for a specific class that implements EPS.
  + Arguments:
    - <required type if applicable> <arg1>
      * <purpose>

**Interface Instance Methods (used on an instance of this class):**

* public <other tags> <return type> <method name>(<#> arguments)
  + Purpose: <1-4 sentences> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque in turpis posuere, pretium velit in, tempor turpis.
  + Priority: <Same options as the class priority> (<reason>)
  + Arguments:
    - <type> <arg1>
      * What to put there (see also: <something else in this file>)
  + Returns: (<type>) <what data is returned>
    - <Can go onto a separate bullet(s) if it would be cleaner>
* public synchronized void setPower(1 argument)
  + Purpose: Adjust the motor power, and start or stop it when necessary.
  + Priority: Very High (cannot start the motor without it)
  + Arguments:
    - int power
      * The power to set the motor to (see also: MAX\_MOTOR\_POWER)
  + Returns: N/A
* public synchronized void stopMotor(0 arguments)
  + Purpose: Stop the motor, syntactically equivalent to ‘setPower(0)’.
  + Priority: Medium (only a shortcut method, but all it needs to do is call setPower(0))
  + Arguments: N/A
  + Returns: N/A

**Interface Static Methods (used without an instance of a class):**

* public static synchronized long getCurrentTime(1 argument)
  + Purpose: Get how much time has elapsed, syntactically equivalent to ‘System.currentTimeMillis() - someothertime’
  + Priority: Low (prevents other programmers from having to do this manually every time)
  + Arguments:
    - long start
      * when this ‘timer’ started (specify 0 to get an initial start time)
  + Returns: (long) the current system time in milliseconds